

Sistemi Informativi:

Progetto della base dati

argomenti

- progetto della Base Dati
- Progetto per database ad oggetti
- Progetto per database ad oggetti/relazionale
- Progetto per database relazionale

Esempi importanti di DB

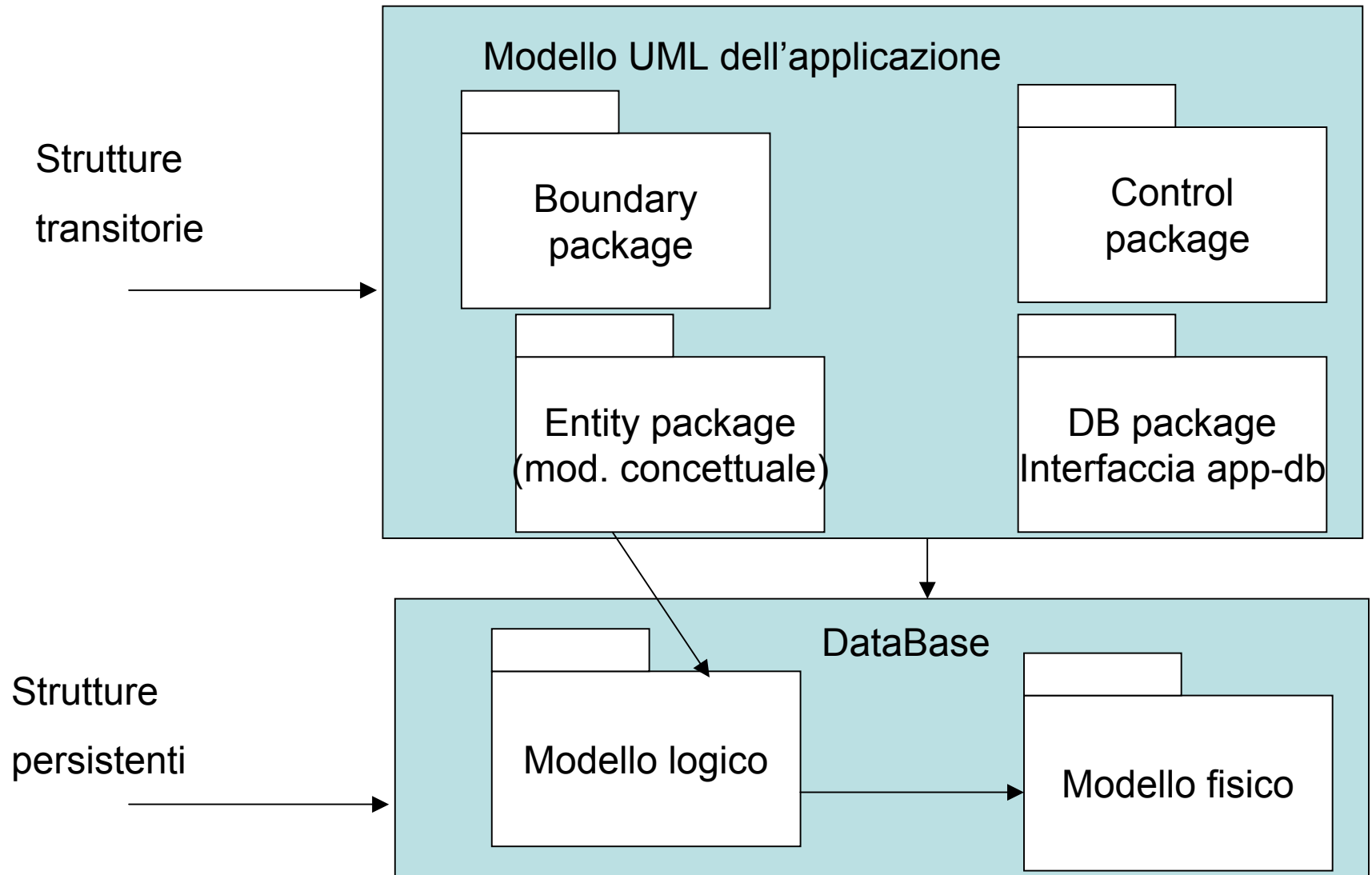
- Oggetti
 - ObjectStore Versant
- Oggetti / Relazionale
 - Oracle8 UniSQL
- Relazionale
 - Oracle sybase DB2 informix Access
- Vecchi modelli
 - Gerarchico (IMS)
 - Rete (IDMS)
 - Adabas
- File normale

Le prestazioni scendono
piu' si sale di astrazione !

Modello dei dati

- Si cerca di fornire modelli a diversi livelli di astrazione per nascondere o mostrare i dettagli
- **Modello concettuale**
 - Modello sintetico che concerne una singola applicazione
- **Modello logico**
 - Tiene conto di tutte le applicazioni presenti e future che accedono a quei dati
- **Modello fisico**
 - Descrive i meccanismi di memorizzazione dei dati sui supporti fisici

Modello persistente ed applicazione



Rappresentazione di oggetti in un DB

- Nella maggior parte dei DB in uso corrente le strutture dati del DB sono molto diverse (piu' semplici) da quelle degli oggetti
- Il DB deve essere progettato per fornire un accesso ottimale a tutte le applicazioni che ne faranno uso
- Tutto sarebbe facile se non ci fosse un grosso problema di efficienza per i DB di grandi dimensioni
- Traducendo un modello UML ad un DB, l'enfasi e'posta sulla modellazione dei dati; vi e' poca enfasi sul supportare vincoli di visibilita'delle operazioni associate alle classi

Il modello di un Object DB

- Standard del Object Database Management Group (ODMG)
- Un ODB fa apparire oggetti del DB come oggetti del linguaggio
- Interfaccia Object Storage - implementa il DB package
- ODBMS – il sistema di gestione di un ODB fornisce in genere un linguaggio Object SQL (OSQL) che permette di navigare un ODB come se fosse relazionale

Tipi di dati in un ODB

- Oggetto: ha un OID es Impiegato
- Letterale: non ha un OID es “Mario”
- Distinzione tra **tipo** del dato (specifica) e **classe** (implementazione)
 - Un tipo di dato Impiegato puo avere diverse classi di implementazione, es Java, C++
 - Permette di separare la specifica tramite interfaccia (non istanziabile) dall'implementazione

Tipi di dati in un ODB

- Atomici (semplici)
 - Alfanumerici, es. nomeLibro
- Strutturati
 - es. Impiegato
- Collezioni e parametrizzati
 - Set<t>, List<t>, es. list<Esame>
- Null
 - Valore speciale quando un qualunque attributo di un oggetto e' equivalente a '0'

operazioni in un ODB

- Un ODB fornisce un insieme di operazioni predefinite per la manipolazione di tipi di base e collezioni:
 - Su data, tempo, intervallo
 - Unione ed intersezione di insiemi, presenza di un elemento in una collezione etc

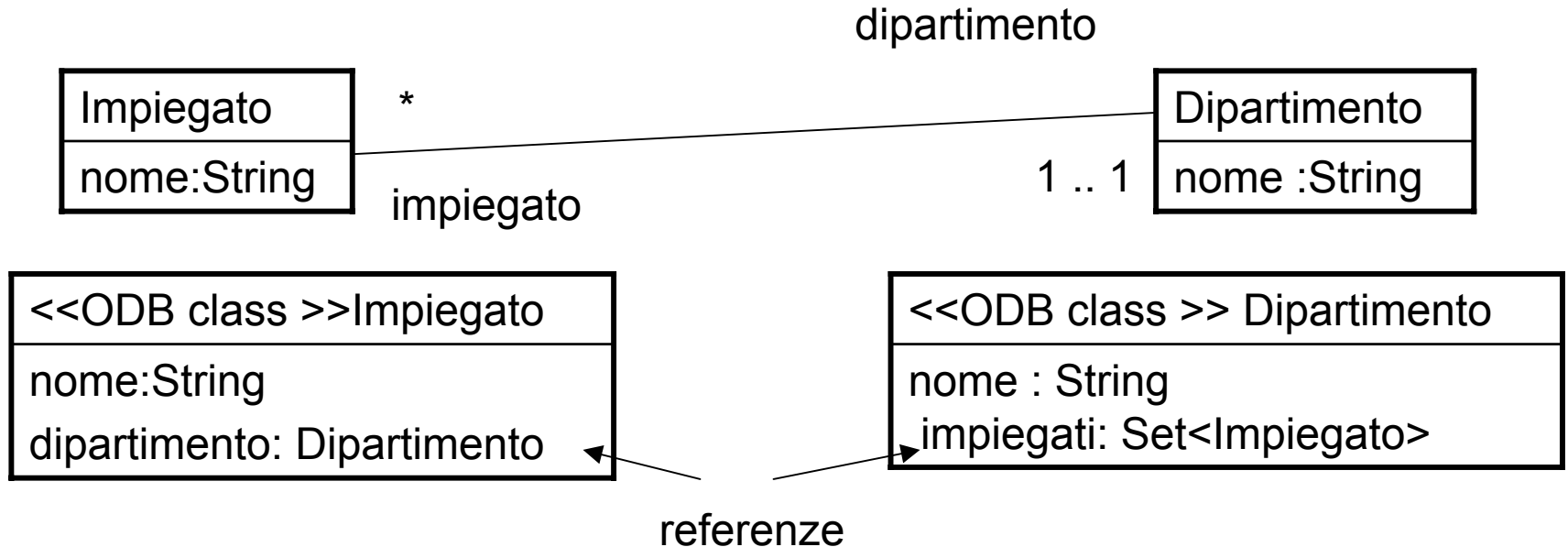
Supporto alle relazioni in un ODB

- Associazioni
 - Implementate tramite collezioni, Set<> e List<>
 - L'aggettivo **inverse** assicura integrita' referenziale
- Generalizzazioni
- Aggregazioni
- Chiavi
 - Semplici e composte per accesso diretto per la iniziale individuazione di un oggetto
 - Diverse dall OID che identifica l'oggetto

Da UML a ODB

- In UML si aggiungono stereotipi dell' ODB
- Si traducono le classi del package **Entity** (oggetti persistenti)
- Si trasforma lo stato degli oggetti, mentre il comportamento e' definito nel progetto architetturale

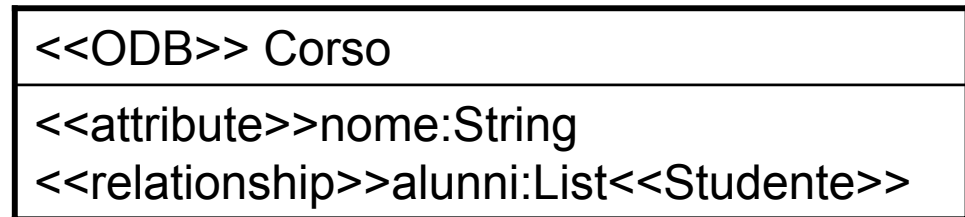
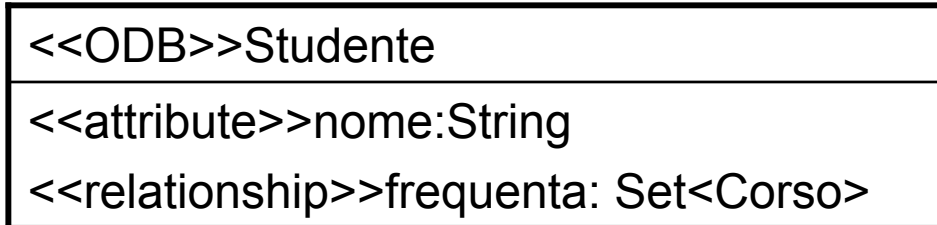
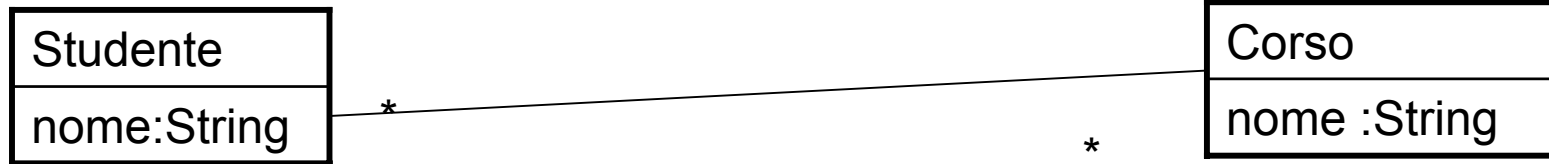
Associazioni da UML a ODB



Se una relazione ha cardinalita' max = 1 l'associazione si mappa con un attributo

Se cardinalita' max > 1 si usa Set (non ordinato) o List (ordinato)

Associazioni in un ODB



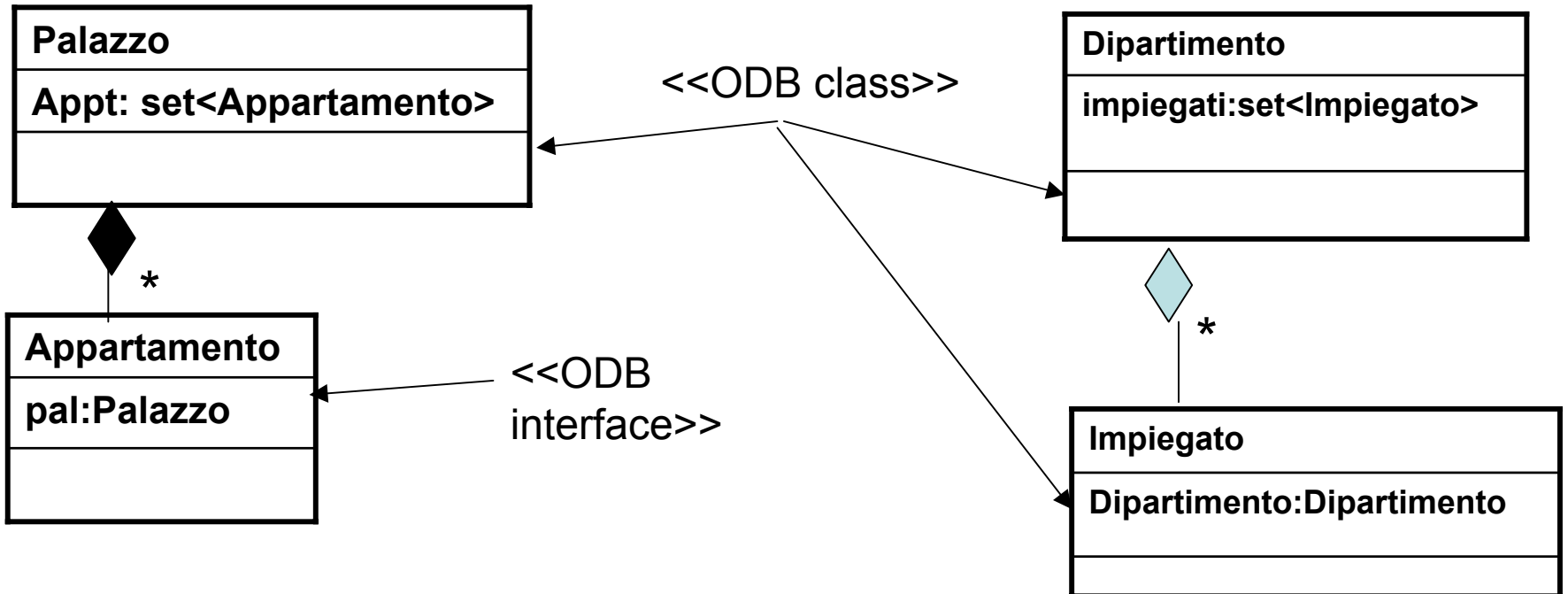
```
Class Studente {
Attribute string nome;
Relationship Set<Corso> frequenta
Inverse Corso::alunni; }
```

```
Class Corso {
Attribute string nome;
Relationship List<Studente> frequenta
Inverse Studente::frequenta; }
```

Mapping di aggregazioni

- Le aggregazioni per referenza si traducono come le associazioni:
- Quelle per valore diventano attributi composti **annidati** nell'oggetto di base.
 - Poiche' non devono essere istanziabili autonomamente, per questi attributi si definisce prima una interfaccia ODB equivalente e poi una classe che la realizza
- La semantica esatta va verificata proceduralmente

Mapping di aggregazioni



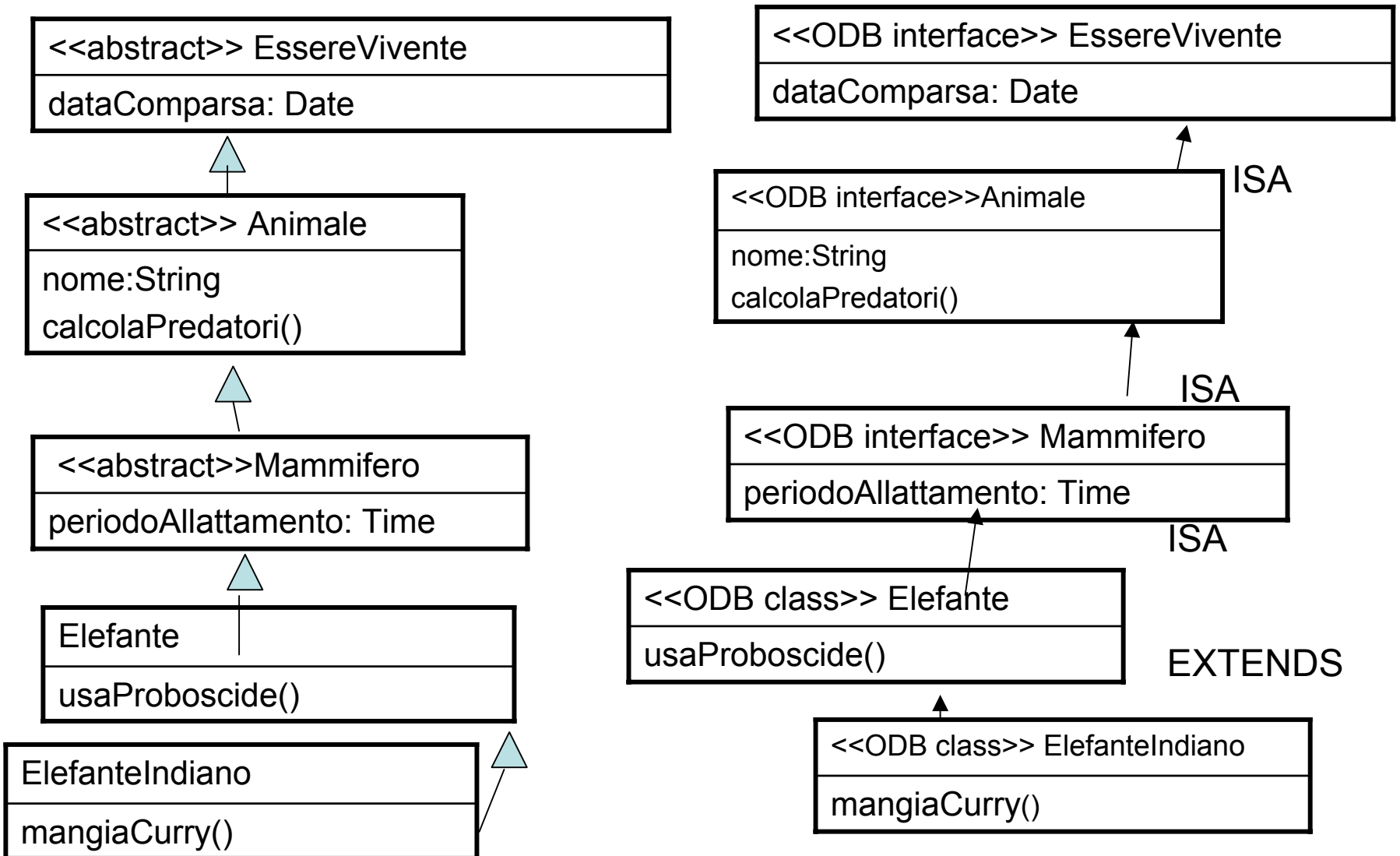
Le aggregazioni si traducono come le associazioni:

Che siano per valore o referenza deve essere verificato proceduralmente

Generalizzazioni da UML a ODB

- **ISA** equivalente ad ereditare interfaccia (comportamento)
 - puo' essere multipla
 - non permessa tra classi
- **EXTENDS** -> ereditare implementazione (anche lo stato)
 - singola
 - Le interfacce non possono ereditare da classi
- Lo standard ODMG
 - Parola riservata (keyword) **interface** definisce una classe astratta
 - La parola **class** per classi istanziabili

ISA ed EXTENDS



Modello ORDB

- Standard ANSI e ISO 1999
- In grado di gestire sia strutture relazionali che strutture ad oggetti
- I tipi di base possono essere composti in strutture arbitrariamente complesse

Tipi in un ORDB

- Memorizzazione tramite tabella
- Tipi definiti dall'utente similmente al modello ODB
 - **Tipo distinto** -> tipo atomico ODB
 - **Tipo strutturato** -> tipo strutturato ODB
 - Attributi (possono essere tipi strutturati)
 - Operazioni: uguaglianza ordinamento, conversioni da un tipo all'altro
 - L'attributo di tipo strutturato puo' essere una collezione: Set, List, Array.....

Dichiarazione di tipo in ORDB

<<structured type>>

FornitoreTY

Id: char(10)

Indirizzo: varchar(125);

Telefono: set (char 20);

Create table Fornitore of FornitoreTY;

Tabelle di oggetti in un ORDB

- Tabella di oggetti- un insieme di righe avente una o piu' colonne
- Ogni riga contiene un oggetto
- Non vi e' definizione di incapsulamento
- Per ogni attributo di un oggetto, il sistema genera automaticamente le operazioni
 - **Mutator – modifica**
 - **Observer - lettura**

Il tipo riga in un ORDB

- Usato per attributi complessi senza dover definire un tipo strutturato appositamente

Create table persona

```
( nome:          varchar(30),  
  indirizzo      row              Simile ai record  
                                strutturati  
    (via:        varchar(30),  
      numero: integer )  
  telefono      row (  
    prefisso: integer,  
    numero: integer))
```

Il tipo **referenza** in un ORDB

- La parola riservata `ref` indica una referenza
- La referenza contiene il valore dell' OID
- Una collezione di referenze implementa una associazione di cardinalita' > 1

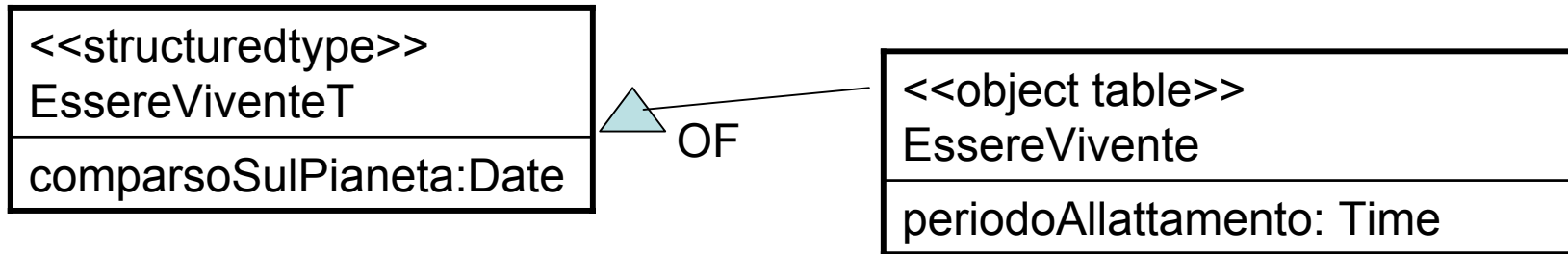
<< <<object table >> Corso
nome:String alunni:list(ref(Studente))

<<object table>>Studente
frequenta: set(ref(Corso))

Colonne , campi, attributi in un ORDB

- La colonna di una tabella puo' essere
 - annullabile
 - valore di un OID
 - una referenza (ref) ad un altro tipo strutturato
- Il **campo** e' il componente di un tipo riga
- L'**attributo** e' il componente di un tipo strutturato
- Campo ed attributo possono essere una referenza

Istanziamento in ORDB



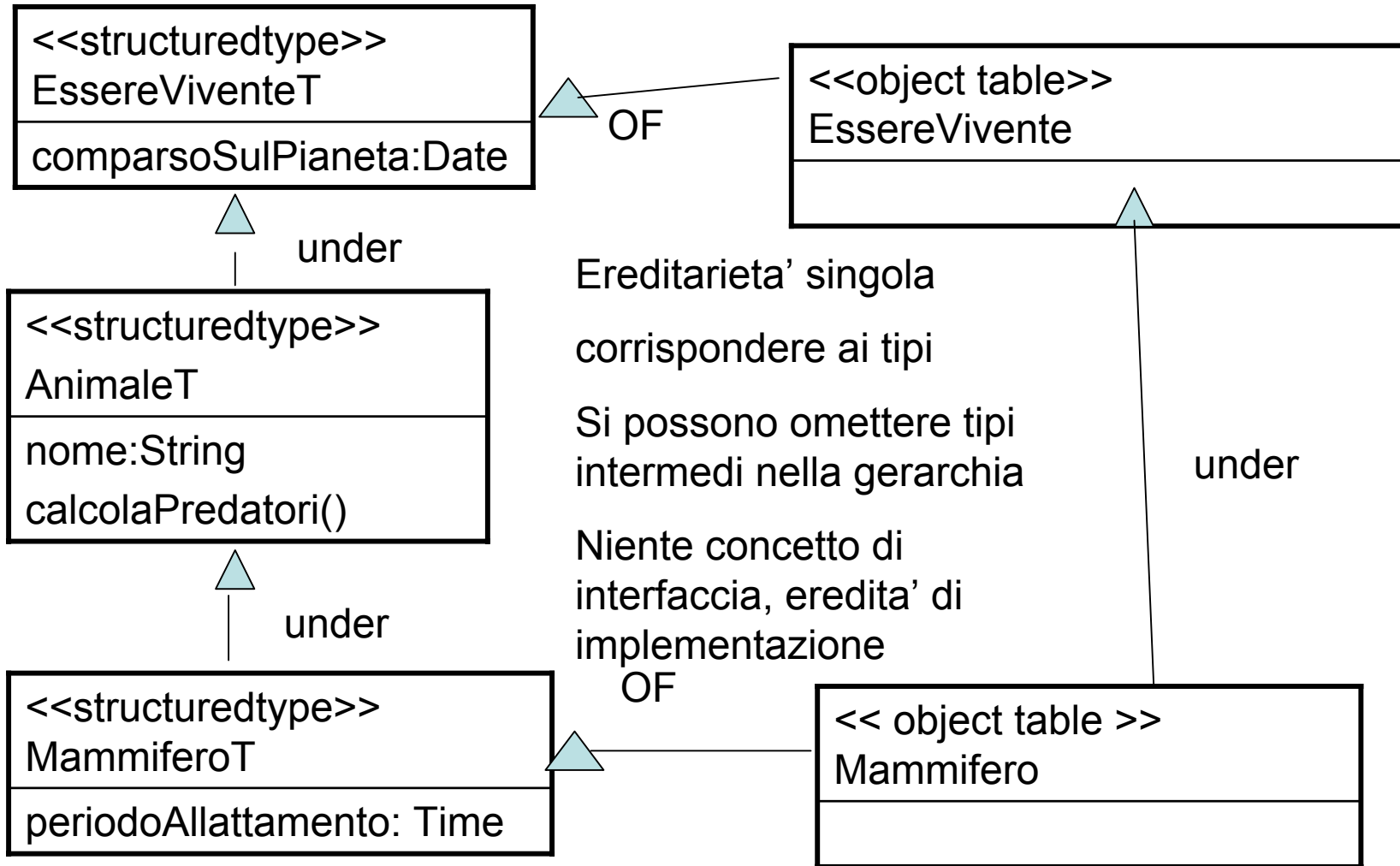
I structuredType sono equivalenti a classi UML; ogni Object table contiene le istanze della classe relativa

Non serve definire una object table per ogni classe se non viene istanziata

Ereditarieta' UNDER

- Ereditarieta' singola
- corrisponde alla generalizzazione
- Niente concetto di interfaccia, eredita' di implementazione

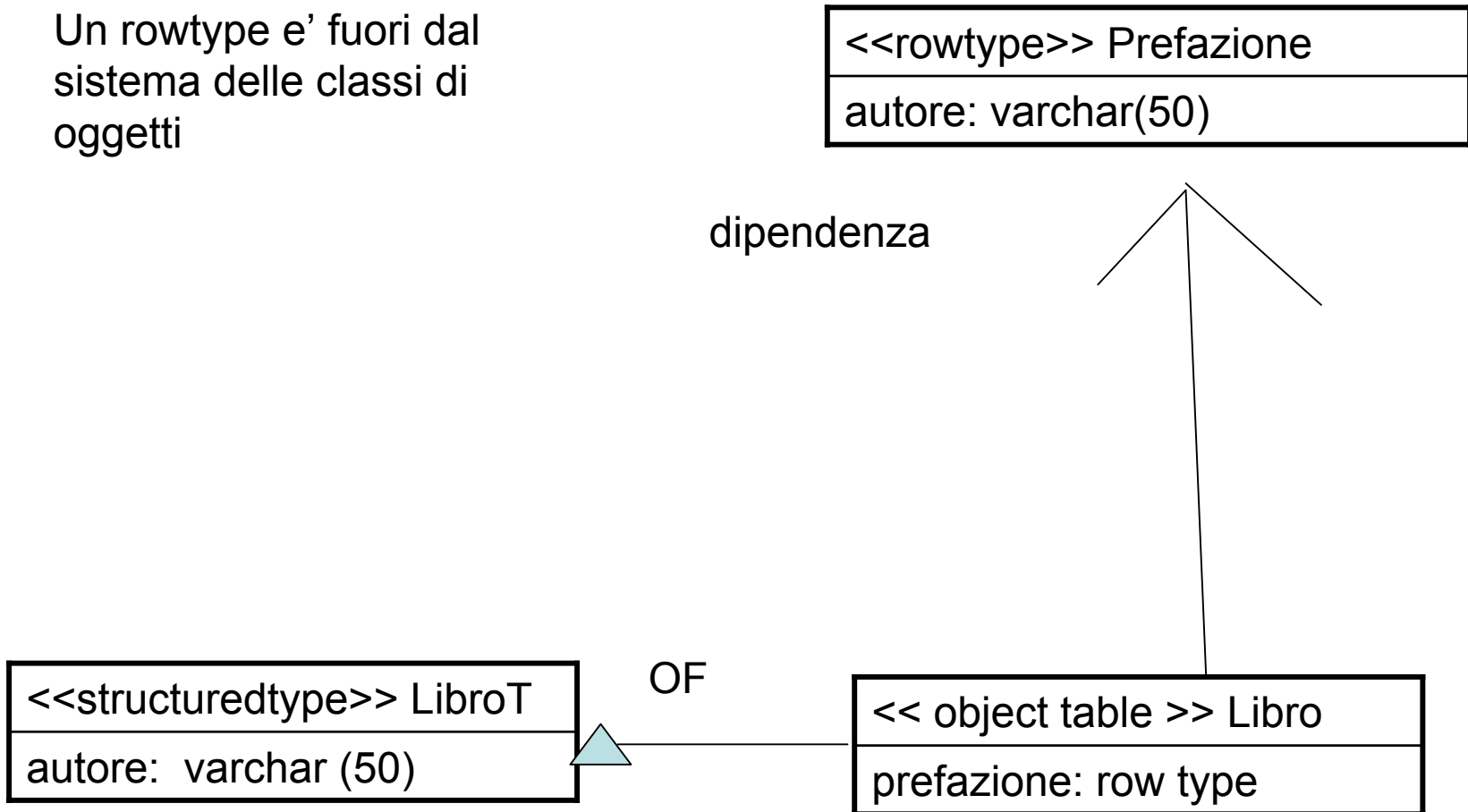
Istanziamento tramite OF ed ereditarieta tramite UNDER in ORDB



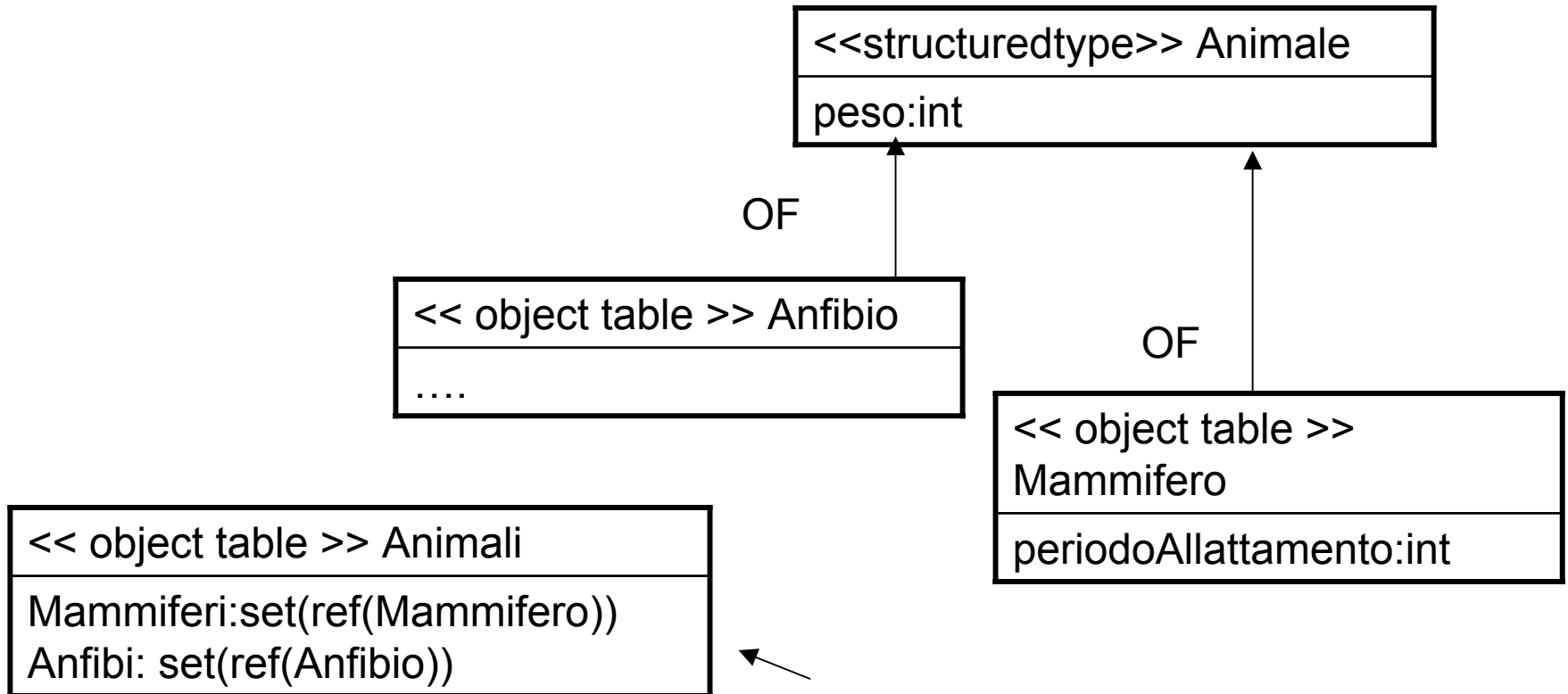
Ereditarieta' singola
 corrispondere ai tipi
 Si possono omettere tipi
 intermedi nella gerarchia
 Niente concetto di
 interfaccia, eredita' di
 implementazione

rowtype e structuredtype in ORDB

Un rowtype e' fuori dal sistema delle classi di oggetti



Mapping di generalizzazioni



Ogni ref puo' puntare ad una sola tabella anziche' ad una qualunque: dunque due insiemi distinti

generalizzazioni UML a ORDB

- Usiamo la relazione OF quando creiamo una tabella che contiene tipi strutturati
- istanziamo degli oggetti
- La parola riservata UNDER denota la ereditarieta'
- La parola riservata FINAL ferma la specializzazione

Create type MammiferoT **under** AnimaleT as (periodoAllattamento:
integer)

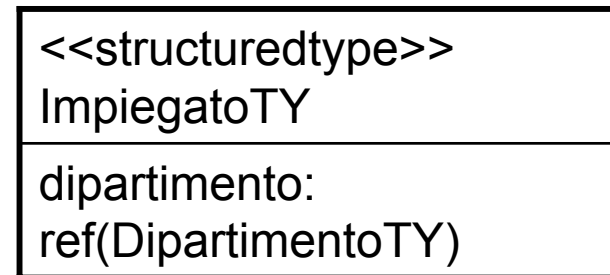
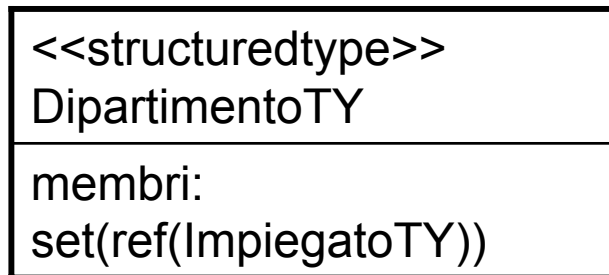
instantiable

final;

Create table Mammifero **of** MammiferoT **under** EssereVivente;

Associazioni e aggregazioni da UML a ORDB

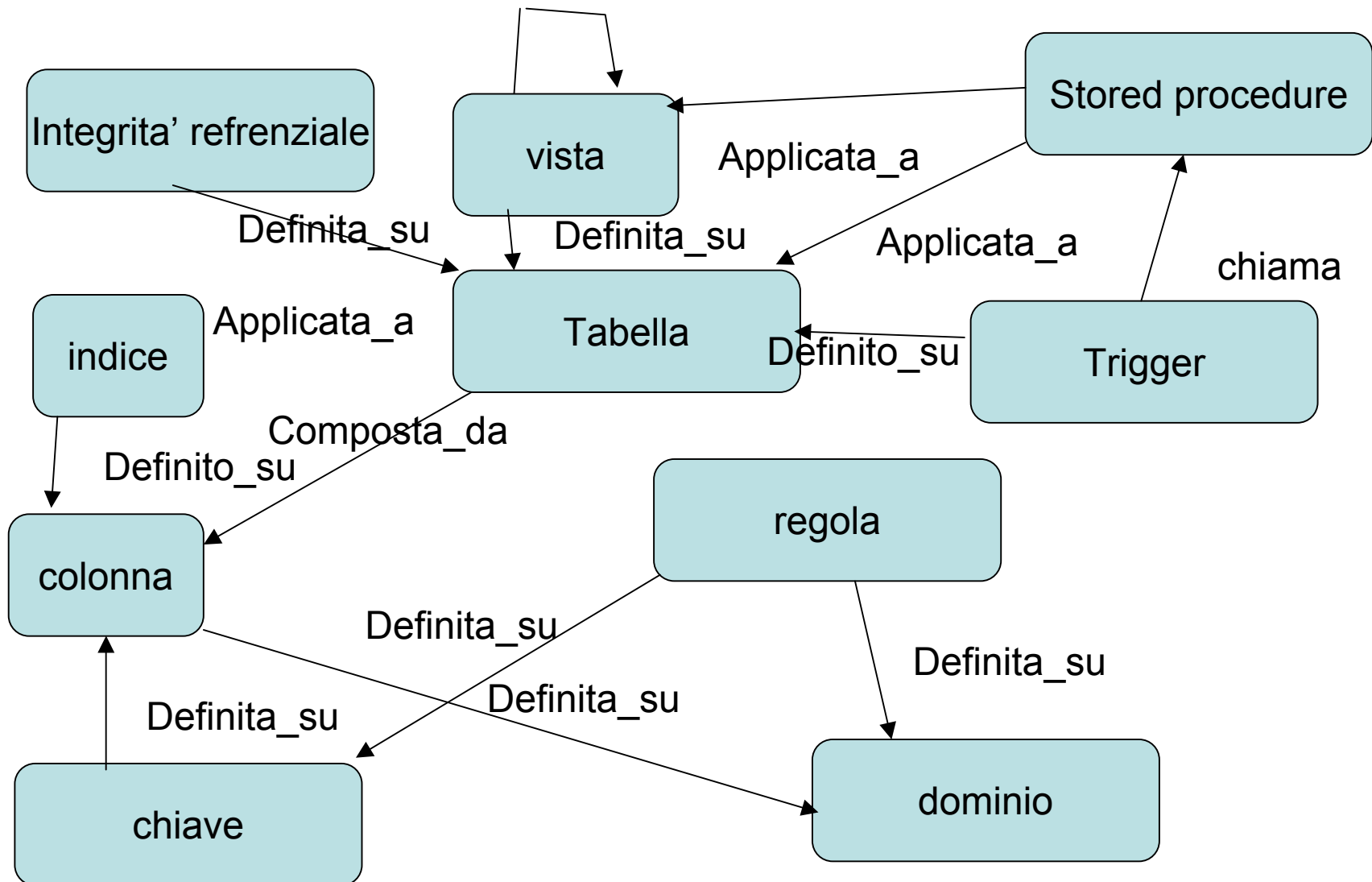
- Tradotte come **ref** (**set** o **list** se multiple)
- Sostanzialmente identico al caso ODB
- Se omettessimo **ref avremmo valori anziche' referenze**



Modello RDB

- Standard ANSI ISO 1992
- Sostituiva modello gerarchico
- Sara' sostituito da ORDB (quando ?)
- La interfaccia Object Storage permette ad un ODB di basarsi su un RDB, con perdita di efficienza

Strumenti di modellazione in RDB



Domini, colonne e regole in RDB

- Dominio - insieme valori possibili per una colonna
- Le colonne memorizzano solo tipi elementari
 - Le regole impongono vincoli procedurali sui valori memorizzabili:
 - Es. la data deve essere gg/mm/aaaa
 - Se la data non e' immessa, assumi oggi

tabella relazionale RDB

- Numero fisso di colonne
- Corrisponde ad un insieme matematico: non ci possono essere due righe identiche
 - Non ci sono dati duplicati
 - Una riga e' identificata tramite un attributo definito **chiave primaria**
 - Insiemi di altri attributi possono essere usati come chiavi **alternative**

Impiegato
codice: Integer <pk>
cognome : varchar(30) <ak>
telefono: varchar(20)
indirizzo : varchar(60) <ak>
....

Integrita' referenziale

- Occorre verificare quando si modifica una tabella che non vi siano righe di altre tabelle che si riferivano ai valori cancellati
- Se ad esempio avessi due tabelle con treni e passeggeri, potrei imporre che un treno puo' essere cancellato solo se non ci sono viaggiatori prenotati

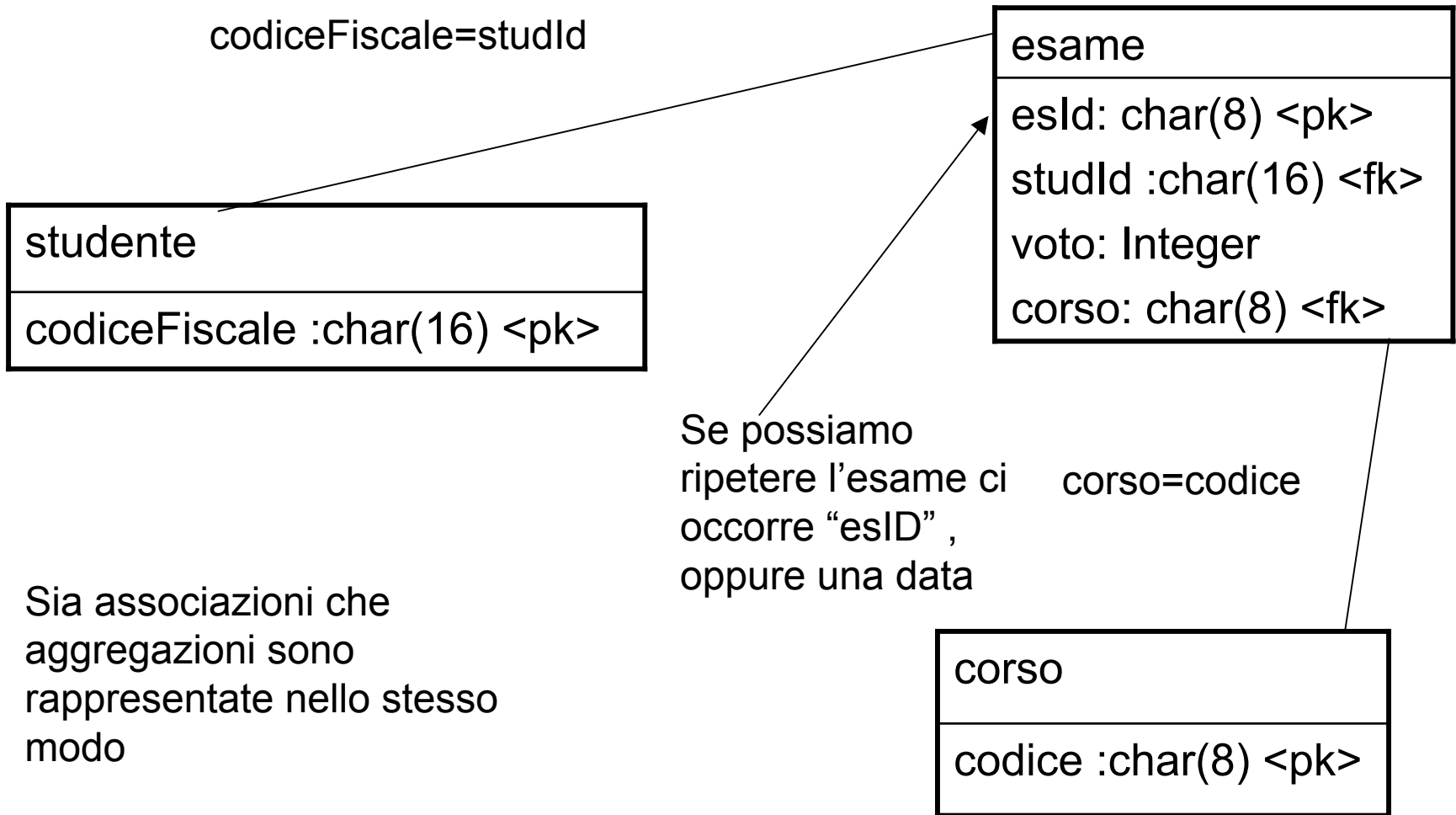
da UML a RDB

- In un RDB non abbiamo referenze, ne' OID.
- Tutte le relazioni che si possono modellare in UML, lo sono in RDB esclusivamente tramite relazioni tra valori delle colonne delle tabelle che fungono da **chiave**
- Due tabelle qualunque possono essere messe in relazione purché esista in entrambe almeno una colonna definita sullo stesso dominio.
- Modello elementare ma che è molto **flessibile** quando si devono fare correlazioni tra dati che non sono state previste in anticipo.

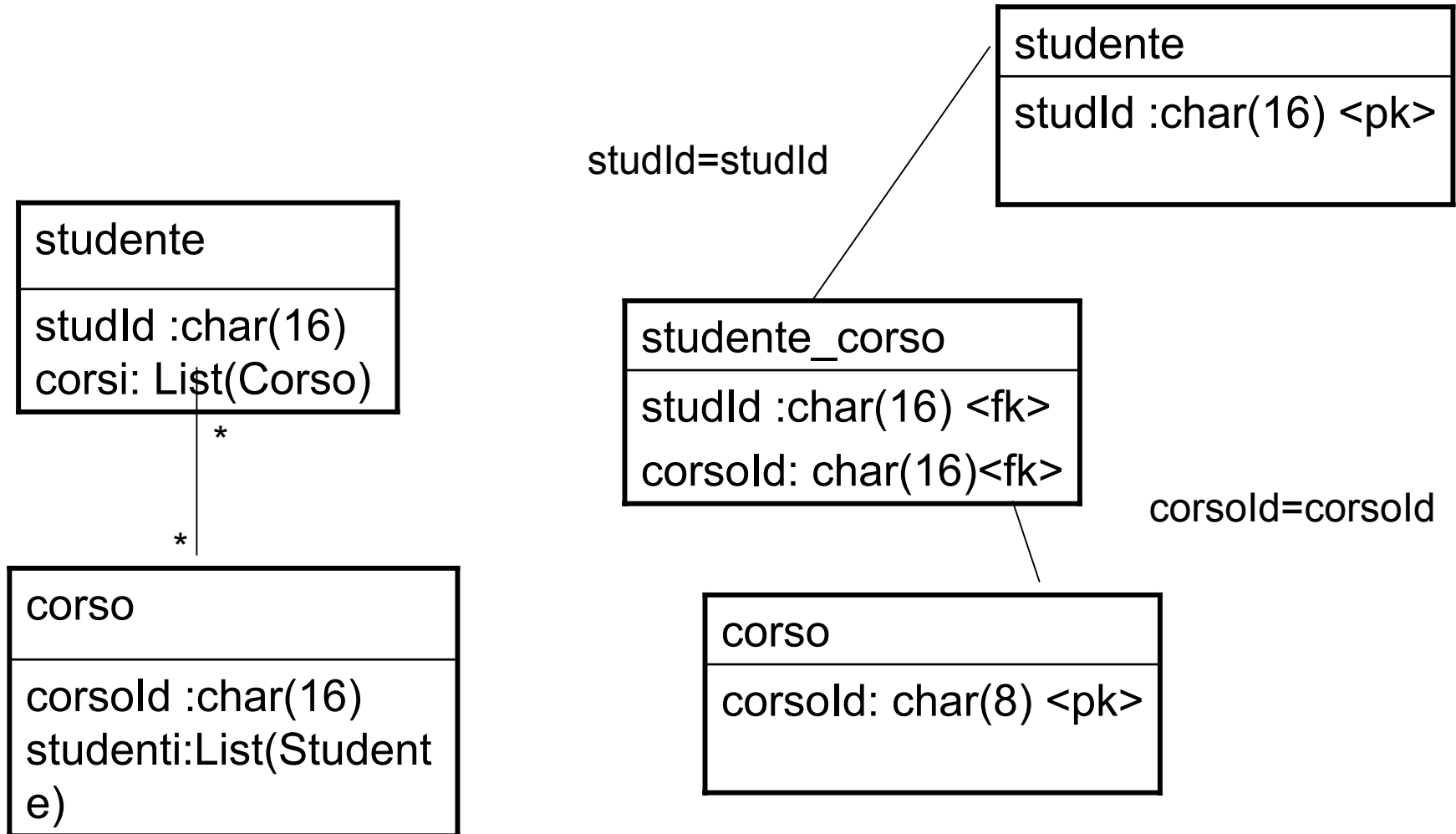
Associazioni ed aggregazioni da UML a RDB

- Associazioni: aggiungere tabelle con chiavi ogni volta che ci sono associazioni card > 1
- Aggregazioni: creare tabelle che memorizzano in ogni colonna una chiave del componente dell'aggregato ;
 - Relazione con la semantica di **member**
- In tutti i casi l'operazione di JOIN ricostruisce l'oggetto desiderato
- Proprieta' rispettate tramite procedure

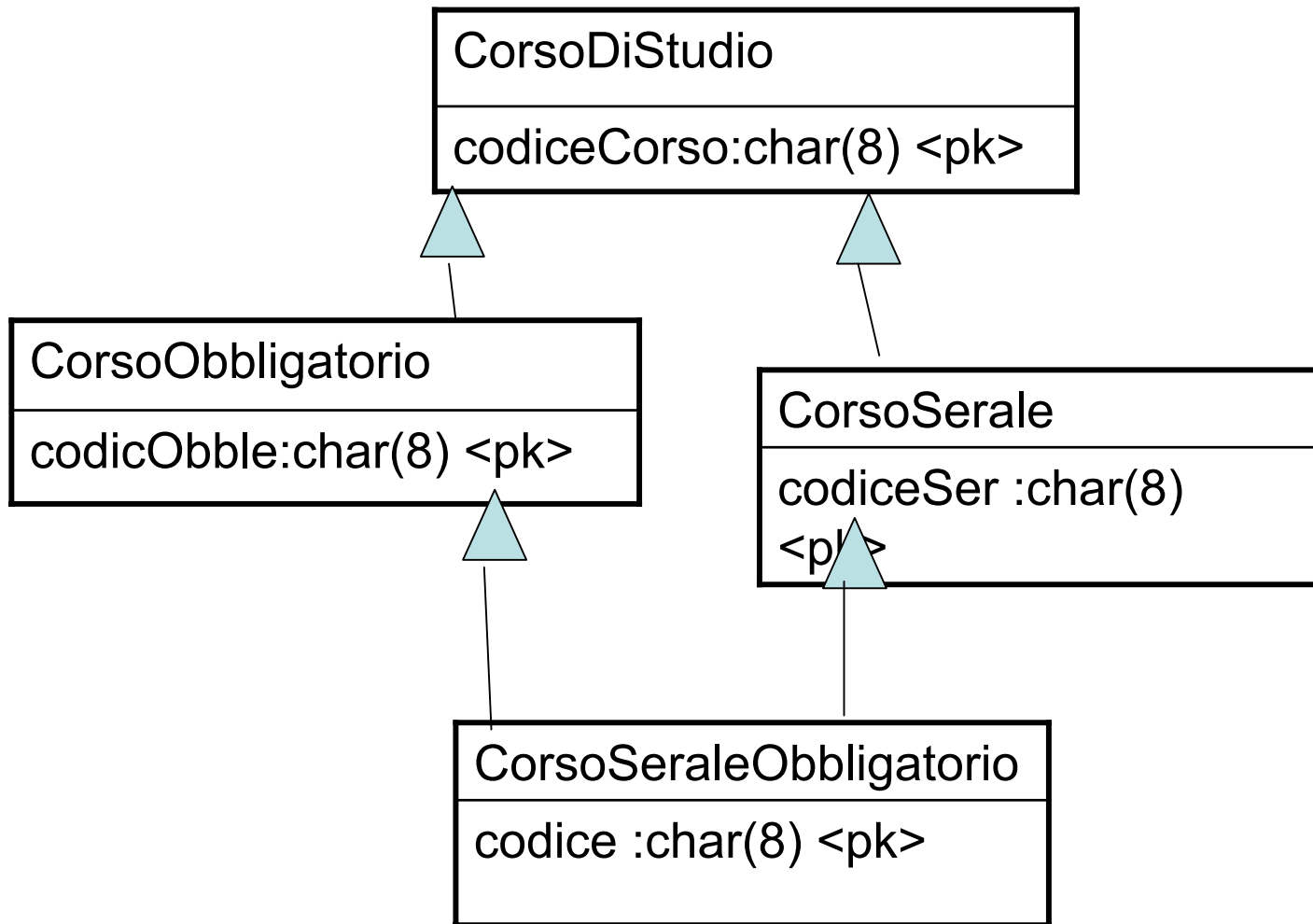
da UML a RDB



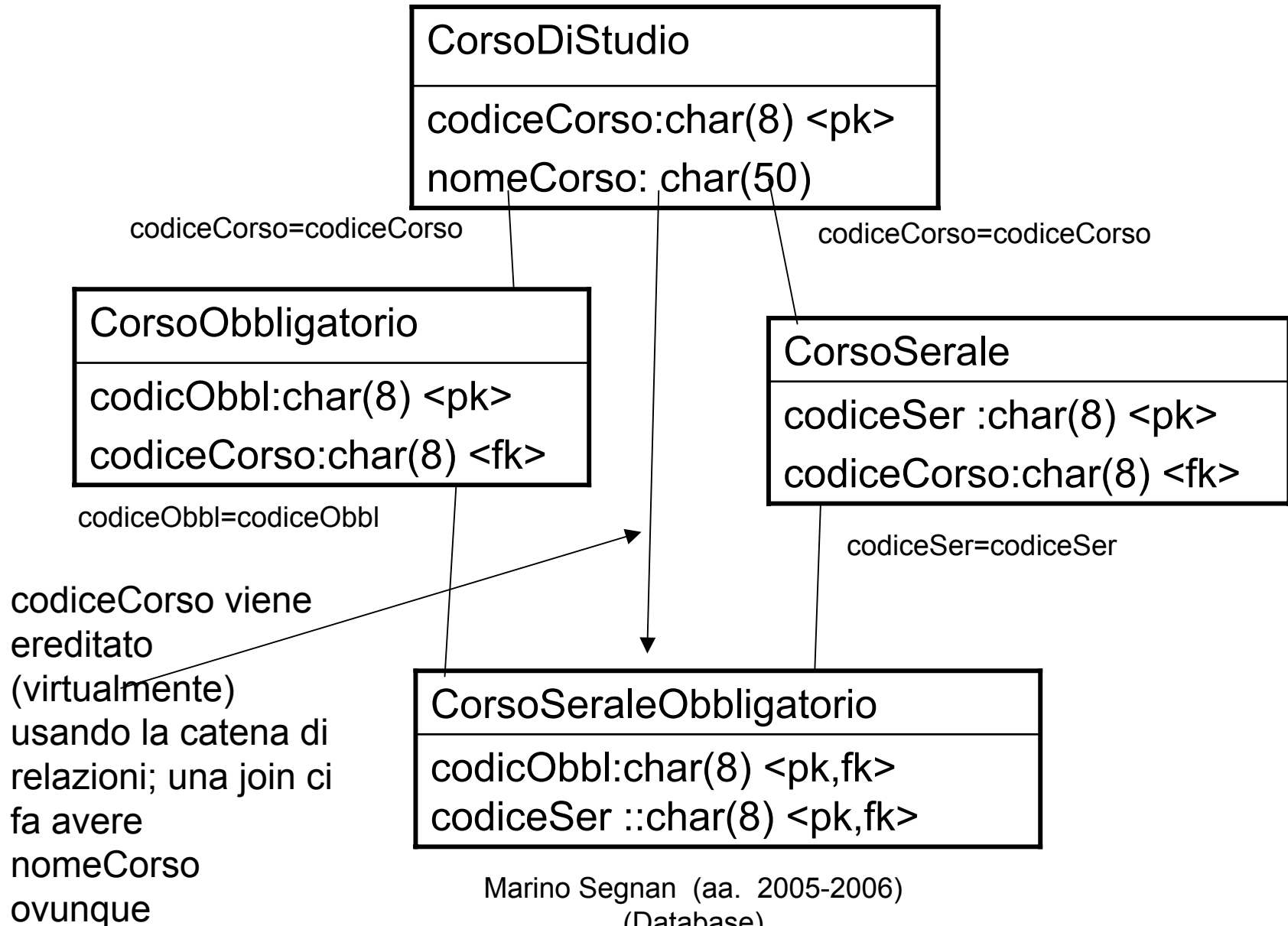
Associazioni multiple da UML a RDB



Generalizzazioni da UML a RDB



Generalizzazioni in RDB 1:1



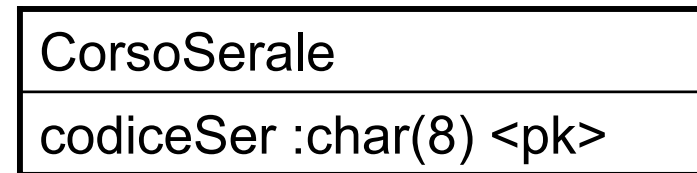
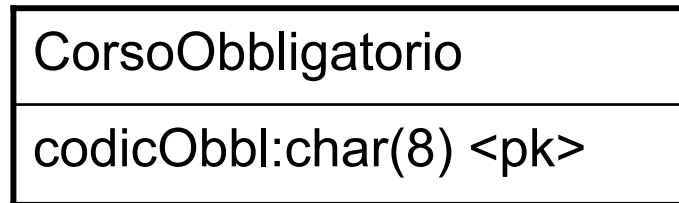
Generalizzazioni in RDB: una classe unica

CorsoDiStudio
codiceCorso:char(8) <pk>
nomeCorso: char(50)
serale: char (1)
obbligatorio: char (1)

Deve poter contenere tutti gli attributi possibili;

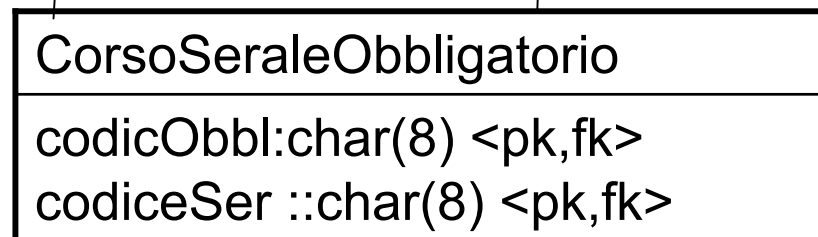
Generalizzazioni in RDB

Ogni classe concreta definita da una tabella



codiceObbl=codiceObbl

codiceSer=codiceSer



Generalizzazioni in RDB

Ogni classe concreta disgiunta definita da una tabella

CorsoObbligatorio
codicObbl:char(8) <pk>
serale: char(1)

CorsoSerale
codiceSer :char(8) <pk>
obbligatorio: char(1)

Sommario

- Tre modelli del DB: concettuale, logico, fisico
- Un modello ad oggetti in UML viene tradotto in modello logico del DB
- La traduzione e' via via piu' difficile passando da ODB a ORDB a RDB:
 - Nel caso RDB in particolare il modellatore decide arbitrariamente la corrispondenza tra oggetti ed entita' del RDB